Procedural Content with SWARM

Approaches

- Interest (especially visual interest) comes from a combination of randomness, and constraints
- Three approaches:
  - Apply **variation** to homogeneous object sets
  - **Distribute** and **Filter**
  - **Random Walks** with **shaping**

Variation

- Take existing homogeneous set of objects (e.g. forest of trees, boulders), add tailored variation
- **-vary** size, colour, orientation
- **select** – choose between certain discrete effects. **frames** - choose texture tiles randomly.
- Simple approach, but works very well to help break up a scene
- Key is rapid iteration to find compelling ranges quickly.

Distribute/Filter

- At heart simple: randomly place particles/imposters/objects over an area with e.g., **source**
- After that, filter, e.g. with maps
  - **mapEmit -range** : only allow in certain bands
  - **mapEmit -density**: probability density function
- Maps can be height maps, or sourced from game

More filtering

- Not just location: **mapEmitColor**
- Use **mapRepel** to push initial distribution around
- Combine maps using basic operations to build up more complex effects
- Source maps from game data

Demo

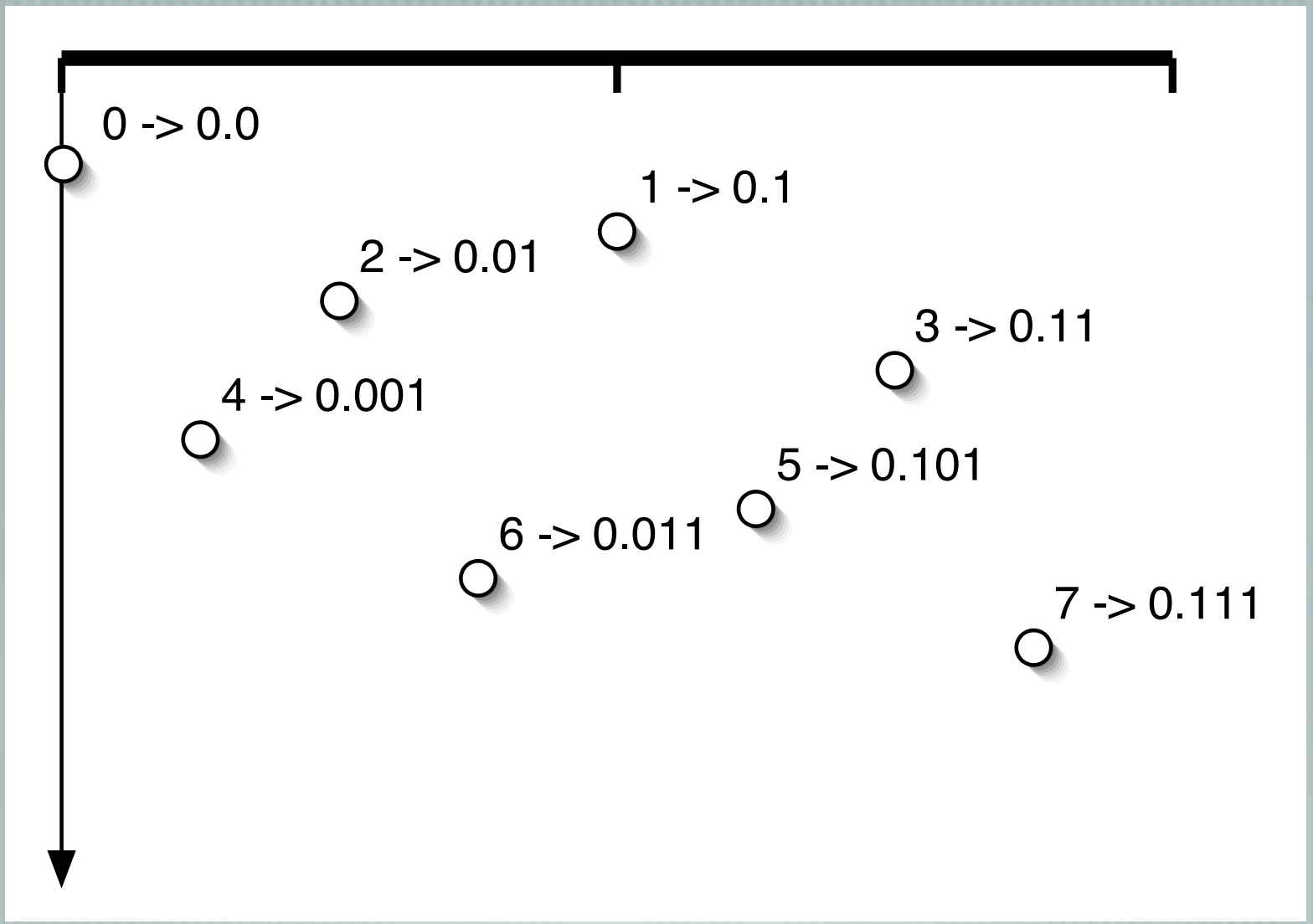
Distribution: Details

- Use rectangular particle source, or source constrained to your domain, just generate x, y, z randomly.
- However, this can have problems:
  - For small numbers of objects, not evenly distributed
  - For large number of objects, clumping
- So use **distribute**

Solution: Halton Sequence

- Pseudo-random number generator
- However, incrementally fills in a line, square, cube, as you take more samples
- Samples are stratified: well sampled
- Can then map to usual set of base shapes: circles, spheres, etc.

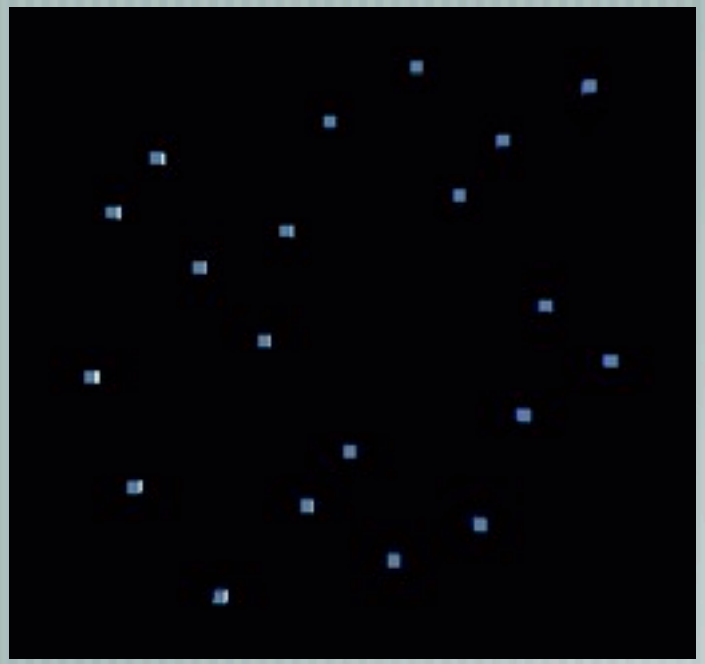
Bit Reversal

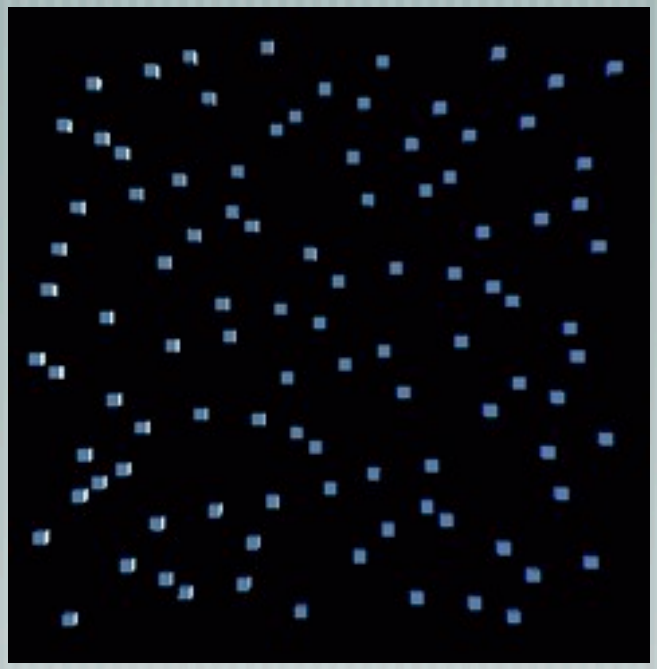


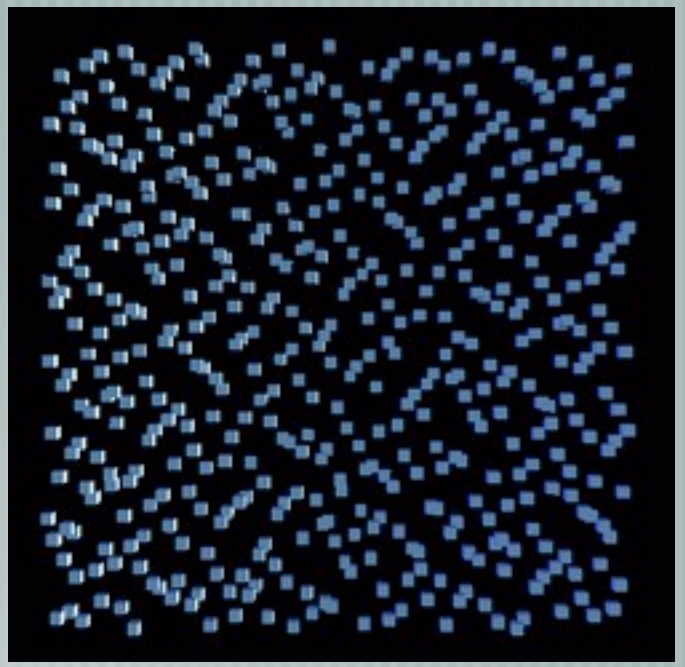
Halton Sequence: But

- Obvious extension to 2D or 3D doesn't work: Points are coherent
- To avoid:
  - Can extend bit reversal to arbitrary base ***n***, where ***n*** is prime
  - Use bases 2, 3, 5 for x, y, z to avoid coherence

Results







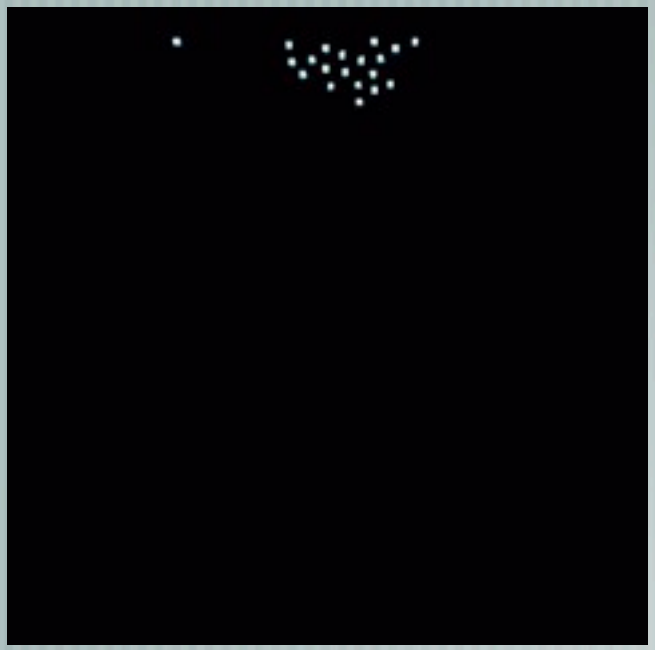
Calculating in Code

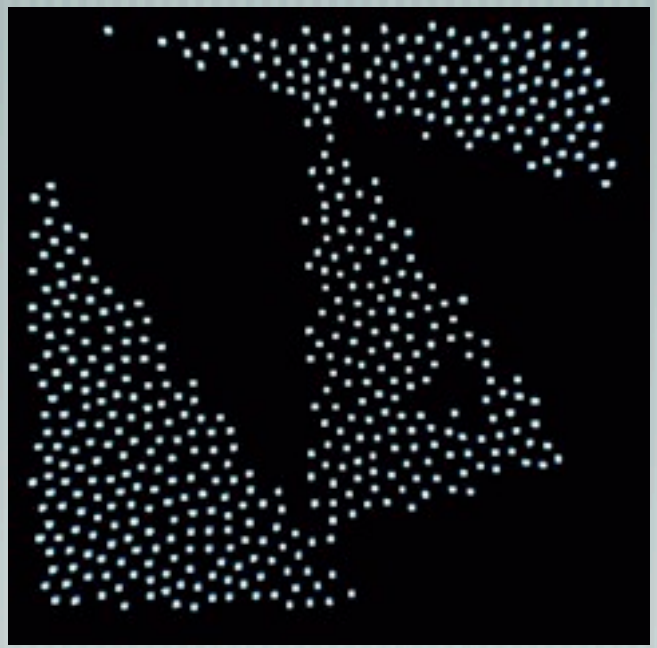
- With a bit of work, can calculate incrementally
  - Completely avoid lookup tables
  - 24 bytes of state
  - More work than an LCG generator, but not a lot more
  - Mostly bit operations, and floating point adds, no divides

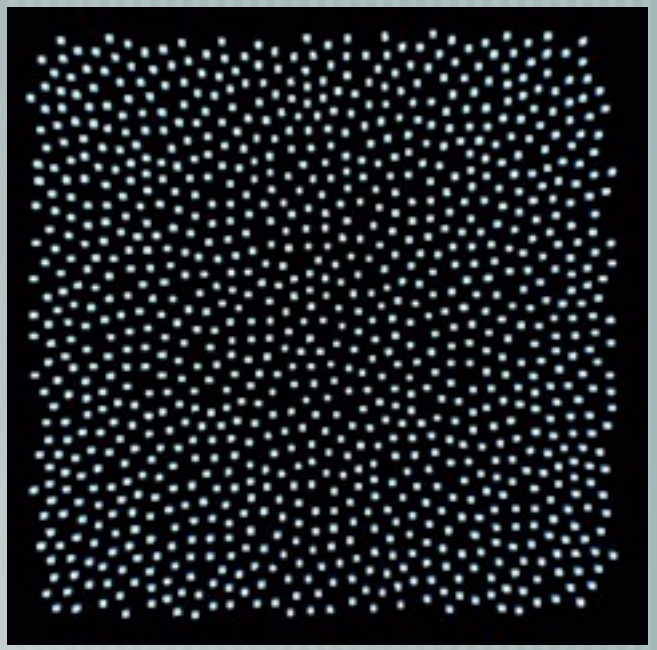
Blue Noise Penrose Tiles

- An alternative: Higher quality stratification
- Not incremental: pick a number of samples, and pre-calculate
- However, pretty fast, especially for less than a thousand samples
- Can modify distribution via density map directly, rather than using Russian Roulette, with good results
- Some memory overhead for generation

Results







Random Walk

- Particle systems have powerful **randomWalk** command – allows variation in types of turns, how far apart turns occur, etc.
- Various options for controlling how model (or entire hierarchical effect) follows that walk: **align**, **-bank**, **-damp**
- In future: more L-systemy walks (script particular sequences of turns over time)

Shaping

- Basic idea: Apply forces to make random walks more plausible
- **force -attractor**: push or pull objects from areas (soccer)
- Maps: **mapRepel**, **mapCollide**
- Generalized surface interaction: surface

Combination

- Forces provide direction to the random walk to make it more plausible
  - Vehicles driving over or above terrain
  - Background creatures, birds, dragonflies
- Combination of randomness and plausible behaviours is powerful

Demo

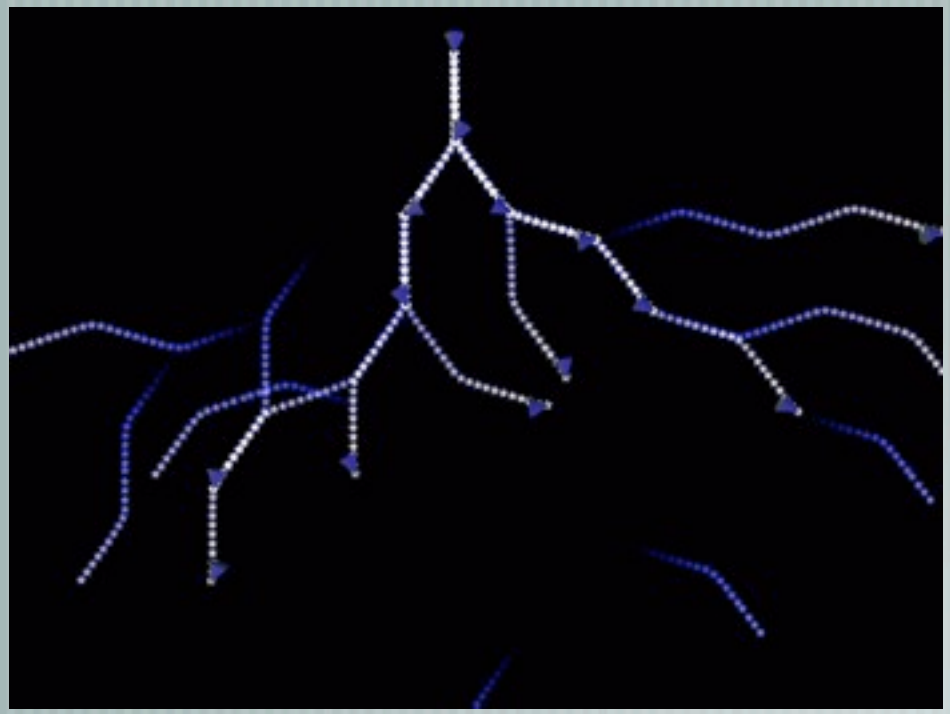
Extensions

- How about applying to objects, walls, creatures?
  - Fur, spots, other features
  - Erosion or dirt simulation
- Easier said than done:
  - Game world is usually straightforward 2D, Z-up
  - Surface of an object is general mesh, local up varies

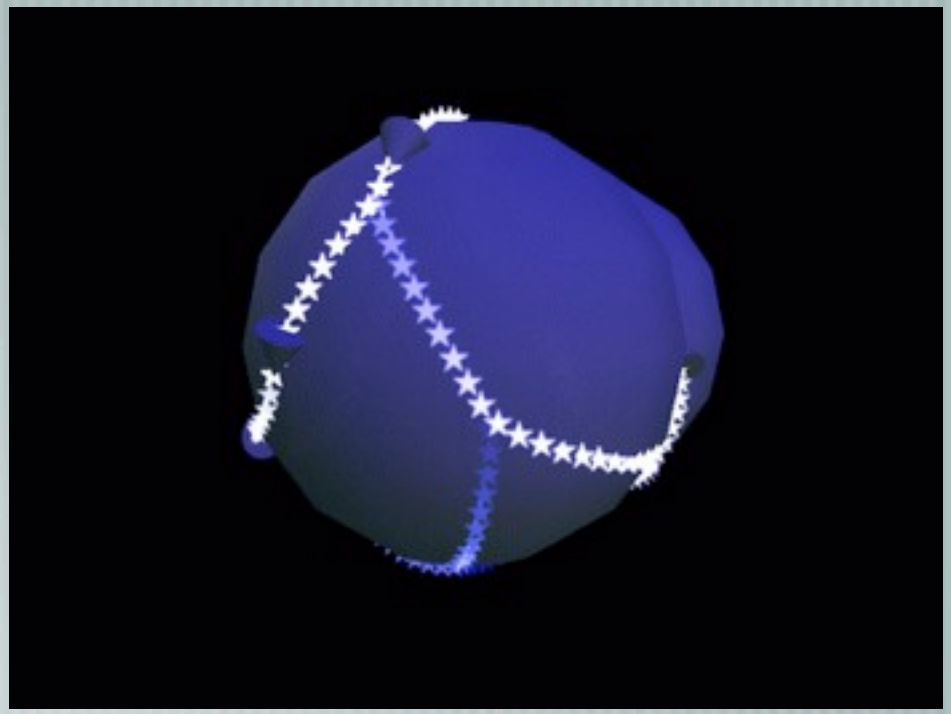
Surface Particles

- Game specifies surface behaviour via interface
- Particle system can drive particles over this with **surface**
  - Random walks
  - Distribution
  - Spray painting!
- Very worthwhile

Surface Particles



Flat



Surface

Closing the Game Loop

- Have procedural stuff feed back into your game world:
  - Lava blob triggers fire simulator event when hitting the ground
  - Explosion sculpts the ground mesh
  - etc.

Further Ideas

- Paths as sources
- Procedural Geometry!
  - Ribbons (particles leave trails behind)
  - Agglomeration: base shapes created, features attached, finally baked into game object mesh
- Chains